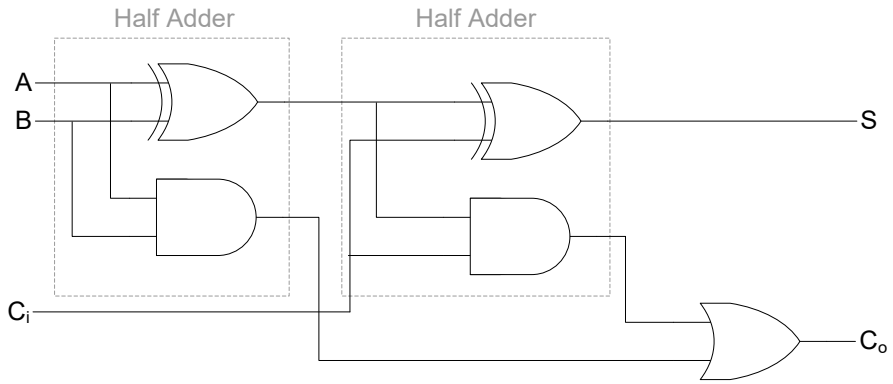
UM EECS 270 F22
Introduction to Logic Design

19. Designing Fast Adders

ripple adder is really slow when bits are larger

Half and Full Adder (Review)



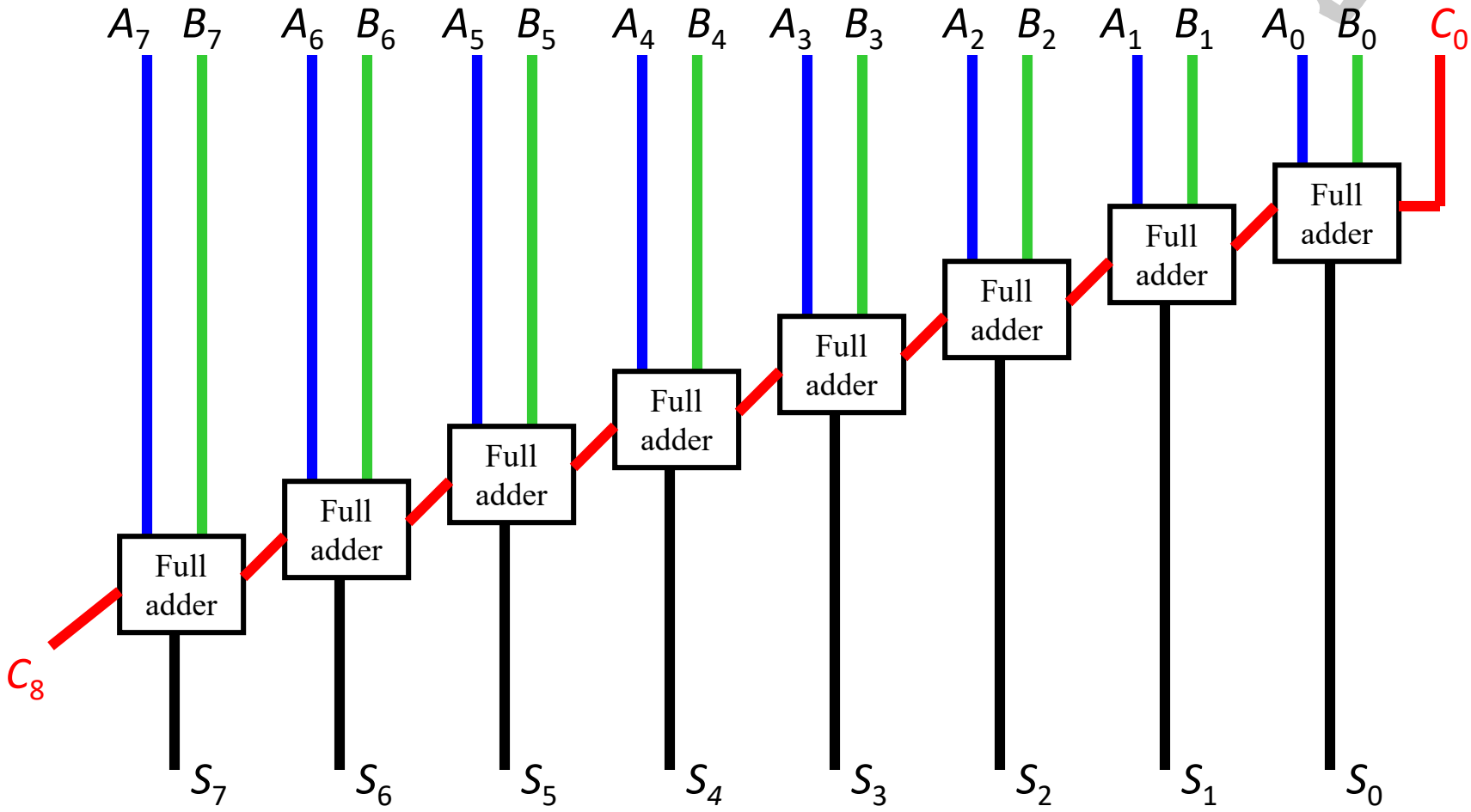
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$C = AB$$
$$S = A \oplus B$$

A	B	C_i	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$C_o = AB + (A \oplus B)C_i$$
$$S = A \oplus B \oplus C_i$$

8-bit Ripple Carry Adder



Unfortunately this has a very large propagation time for 32 or 64-bit addition

General Structure of a Carry Lookahead Adder

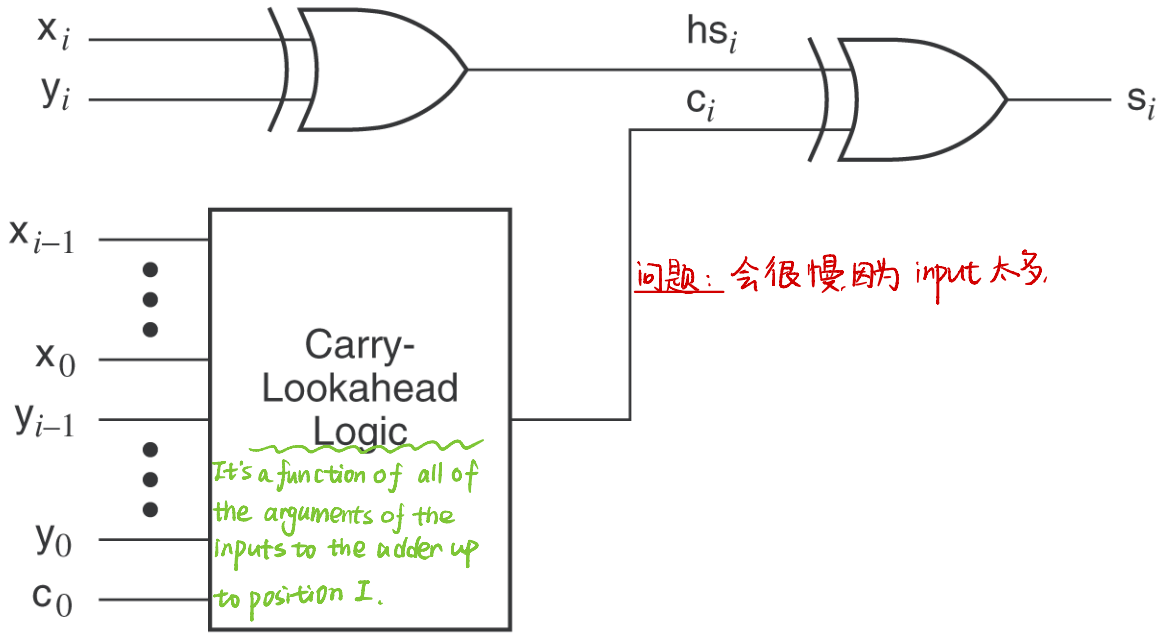


Figure 6-86

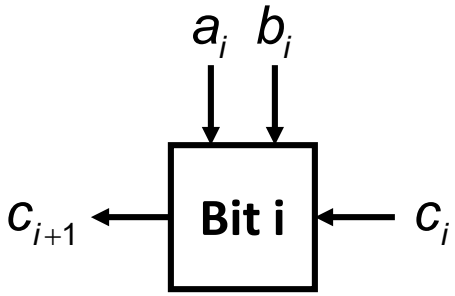
Structure of one stage of a carry-lookahead adder.

Elaboration of CLA Structure

- Bit generate and propagate
- Group generate and propagate
- Carry lookahead across more than one bit
- Recursive computation of group signals
- Carry computation and propagation
- Delay analysis

我们要在 Area 和 delay 中trade off 平衡. ripple adder delay 太大. 上述 PPt Area 太大.

Bit Generate & Propagate



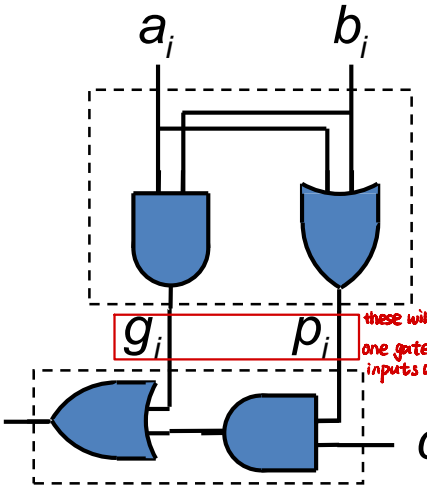
$$c_{i+1} = a_i b_i + c_i(a_i + b_i)$$
$$= g_i + p_i c_i$$

$$g_i = a_i b_i$$

$$p_i = a_i + b_i$$

way one:
Why we call g_i generate signal:
- if g_i is one, output is one. regardless of whether we have a carry in or not.
- we would generate carry out if g_i is true (a_i and b_i are one)

way two:
Why we call p_i propagate signal:
$g_i = 0$. either a_i or b_i is one.
But $a_i + b_i = 1$, and there is a carry in.

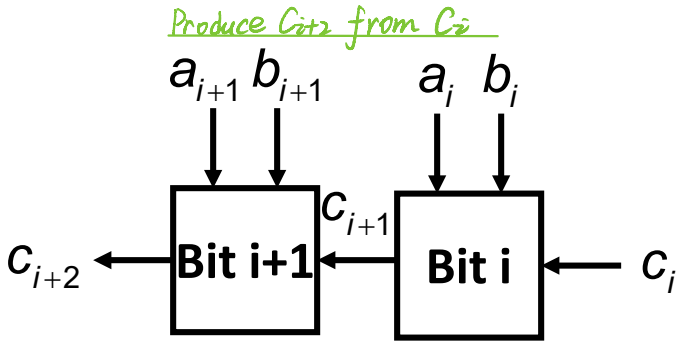


these will happen one gate delay after the inputs are applied to the adder.

Add it will happen across all bits

- g_i gate tells: No matter what is the carry in there's gonna be a carry out here
- p_i gate tells: we will propagate a carry out if the carry in is true.

Group Generate & Propagate: 2 Bits

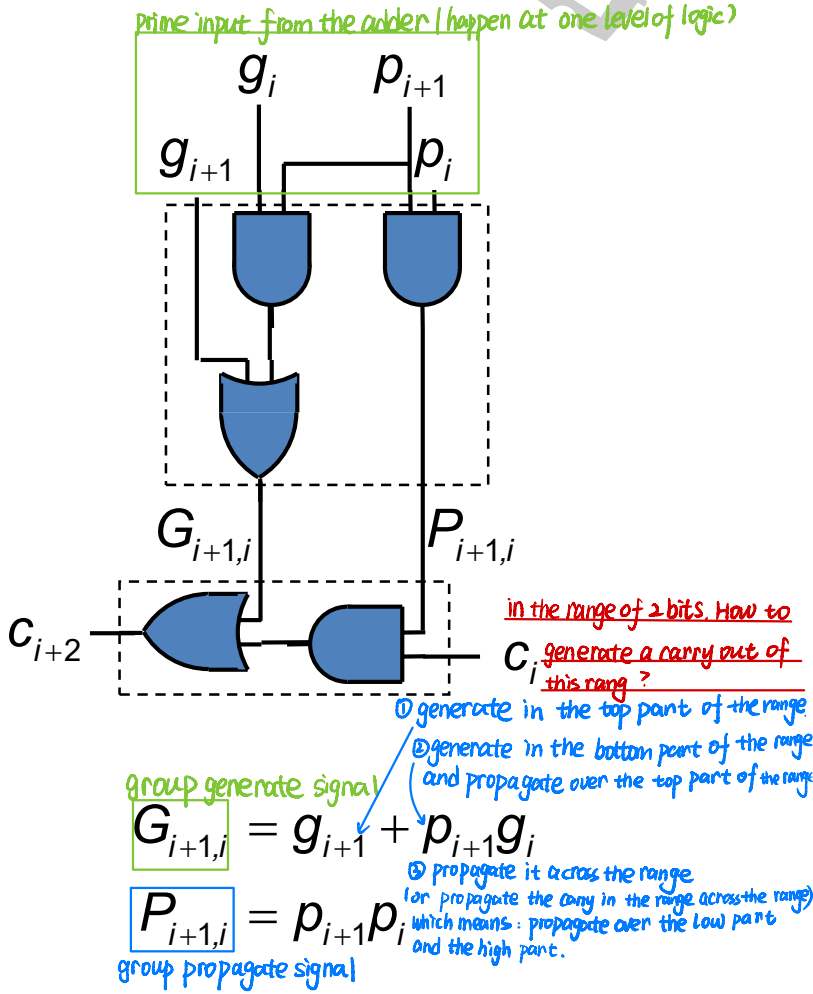


$$
\begin{aligned}
c_{i+2} &= g_{i+1} + p_{i+1}c_{i+1} \\
&= g_{i+1} + p_{i+1}\left(g_i + p_i c_i\right) \\
&= g_{i+1} + p_{i+1}g_i + p_{i+1}p_i c_i \\
&= \boxed{\left(g_{i+1} + p_{i+1}g_i\right)} + \boxed{\left(p_{i+1}p_i\right)}c_i \\
&= \boxed{G_{i+1,i}} + P_{i+1,i}c_i
\end{aligned}
$$

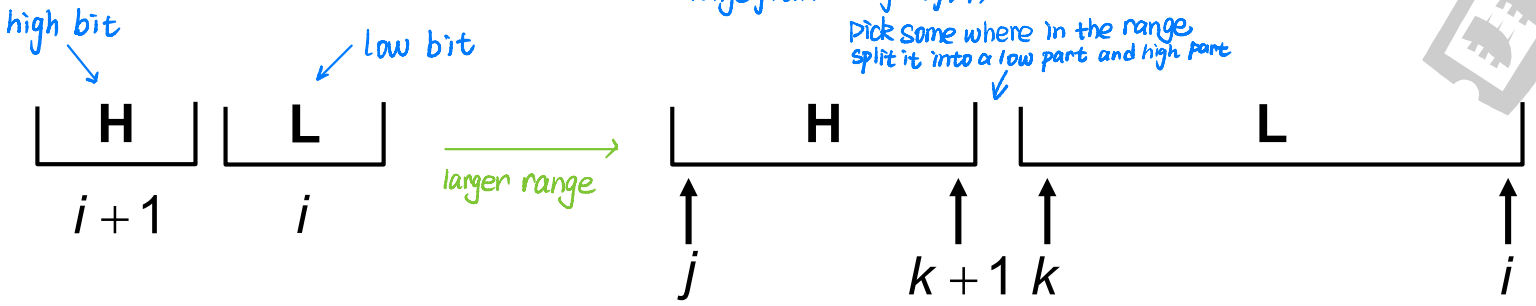
group terms have g together

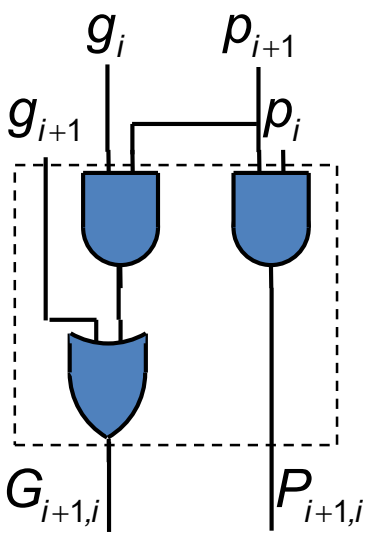
group terms have p together

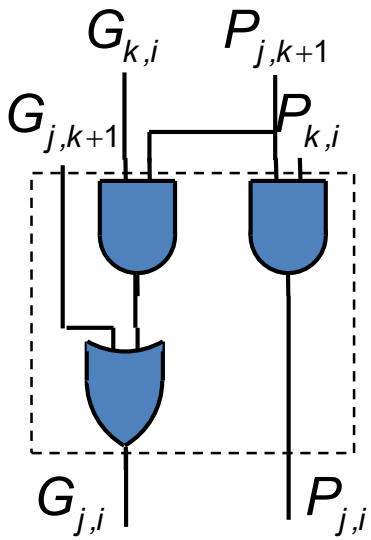
group generate over the bit range from i to i+1



Group Generate & Propagate: >2 Bits







得ijk换成Hi

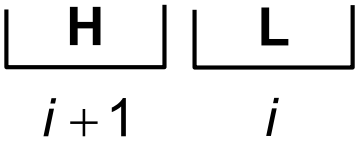
$$G_{i+1,i} = g_{i+1} + p_{i+1}g_i$$

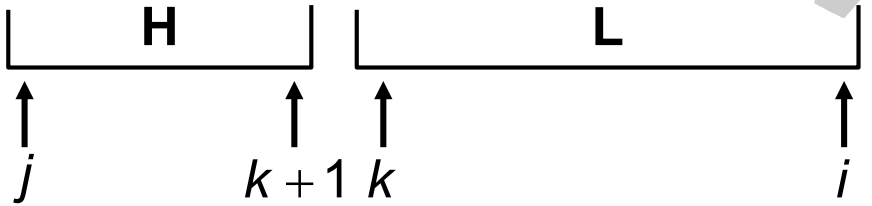
$$P_{i+1,i} = p_{i+1}p_i$$

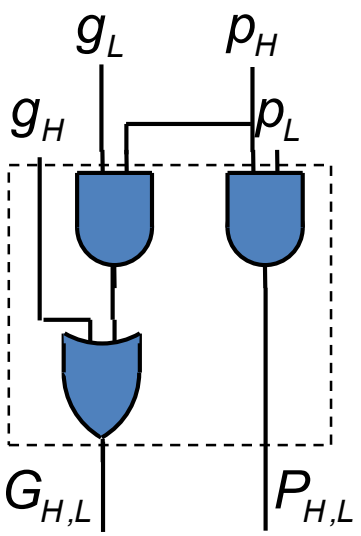
$$G_{j,i} = G_{j,k+1} + P_{j,k+1}G_{k,i}$$

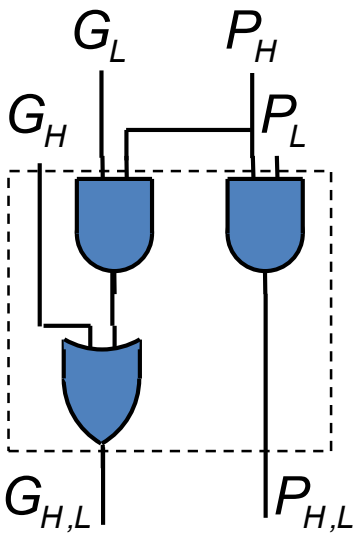
$$P_{j,i} = P_{j,k+1}P_{k,i}$$

Group Generate & Propagate: >2 Bits









$$G_{H,L} = g_H + p_H g_L$$

$$P_{H,L} = p_H p_L$$

$$G_{H,L} = G_H + P_H G_L$$

$$P_{H,L} = P_H P_L$$

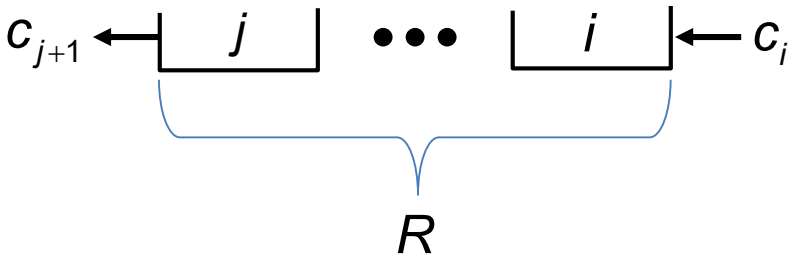
新方法的原因：

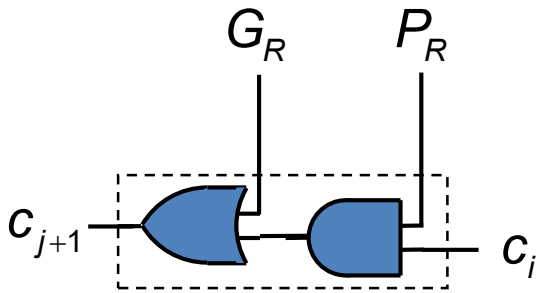
We can anticipate the carry out of the range without waiting for the carries in the middle of the range to be computed.

Carry Lookahead Across Bit Range R = [*i*, *j*]

What we want.

from range "i" to "j", know "C_i", gets "C_{j+1}"





Carry equation:

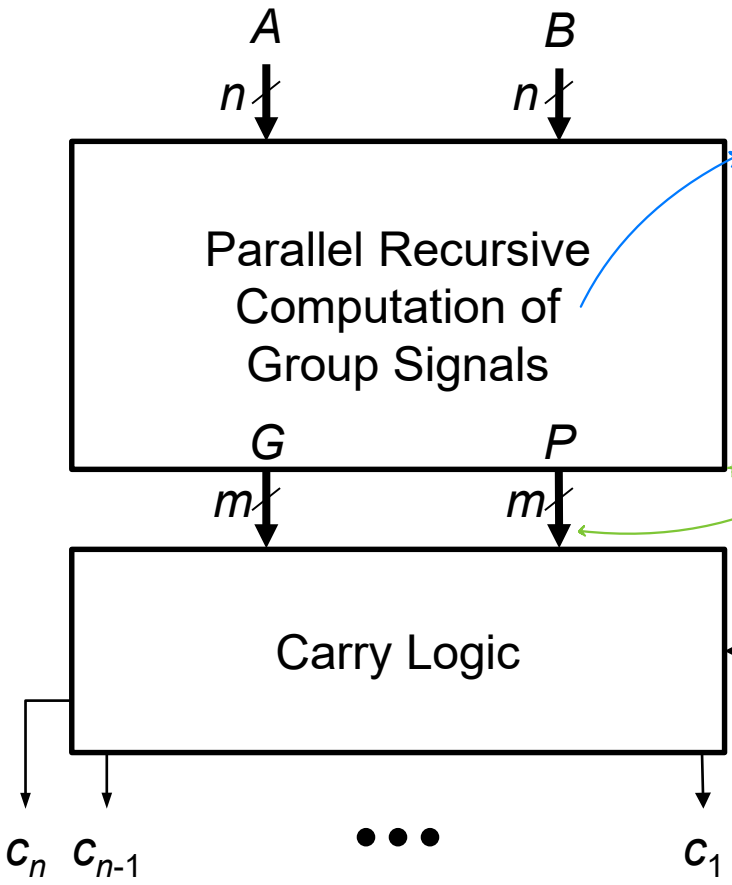
$$c_{j+1} = G_R + P_R c_i$$

get the carry out

if I generate in the range

it propagate across the range.

Parallel-Prefix Circuit



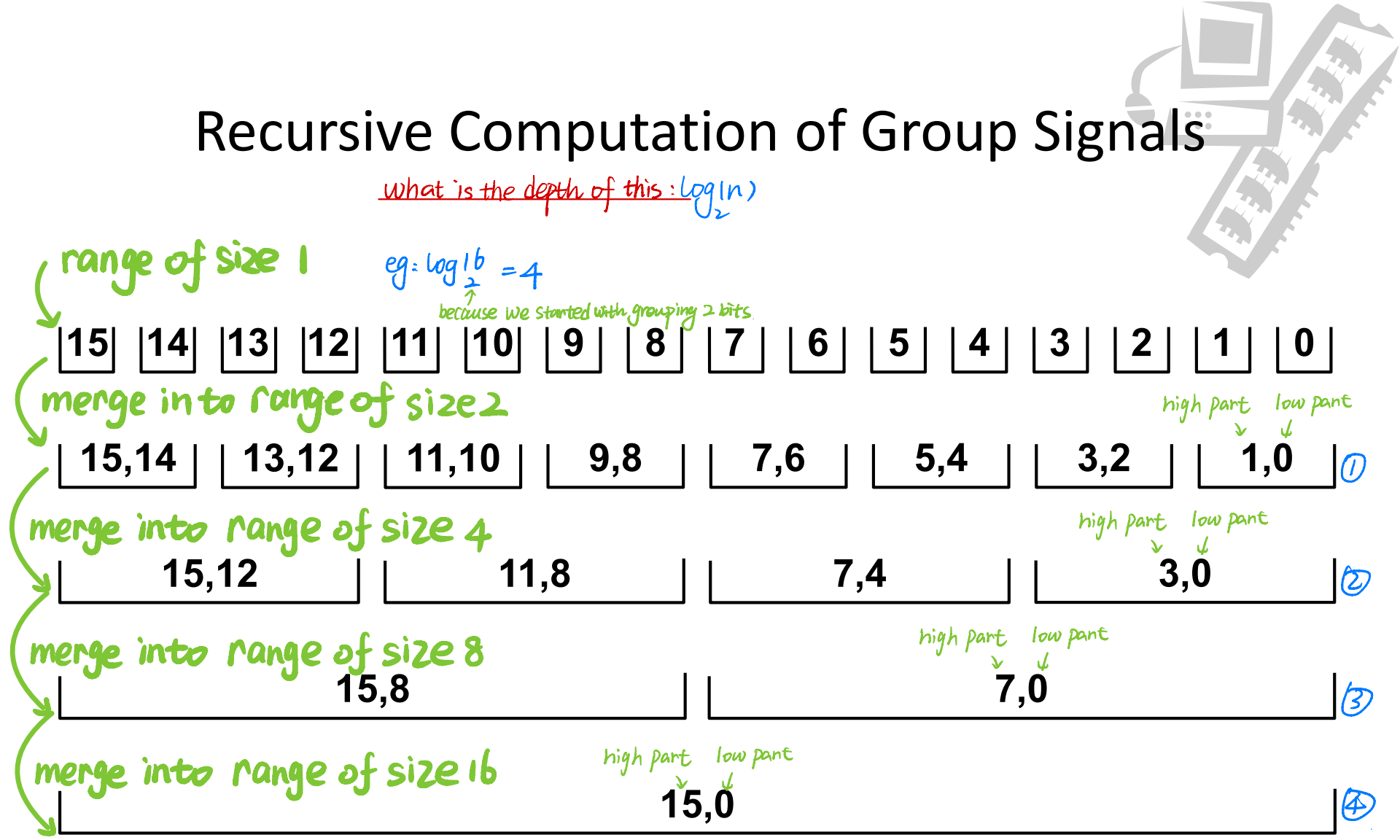
a circuit that produces the propagate and generate across many bit ranges.

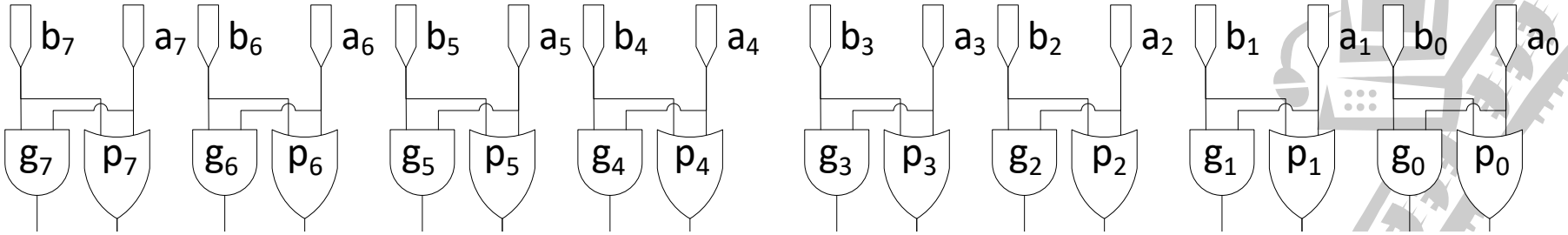
then use them in order to produce the carries.

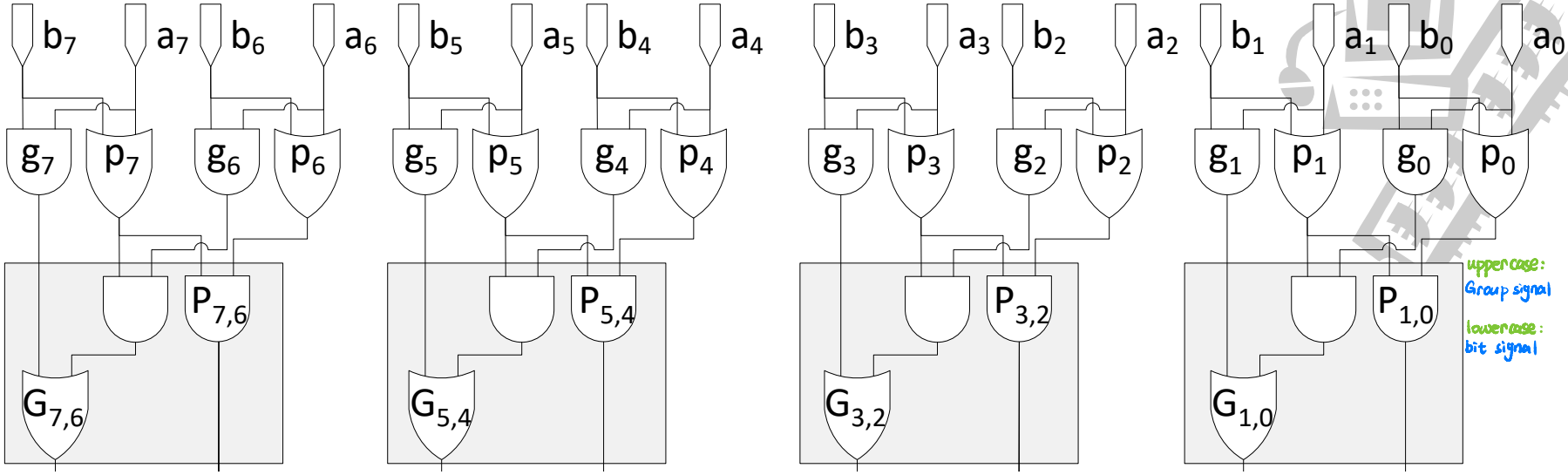
So in this structure: the carries might not come out in sequence. (不按顺序).

回忆，对于 ripple adder: C_1 comes from C_0, C_2 comes from C_1, ⋮

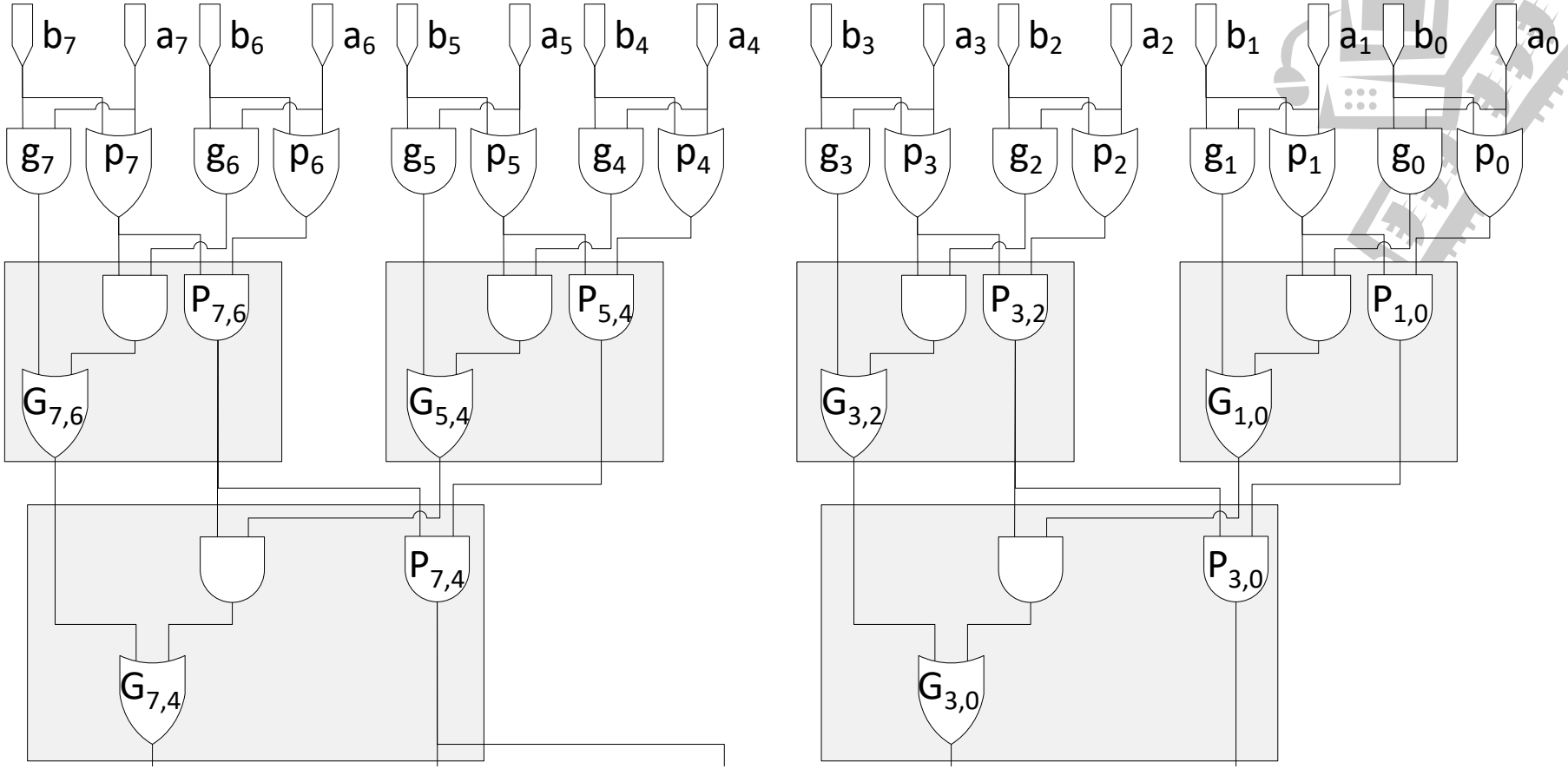
Recursive Computation of Group Signals



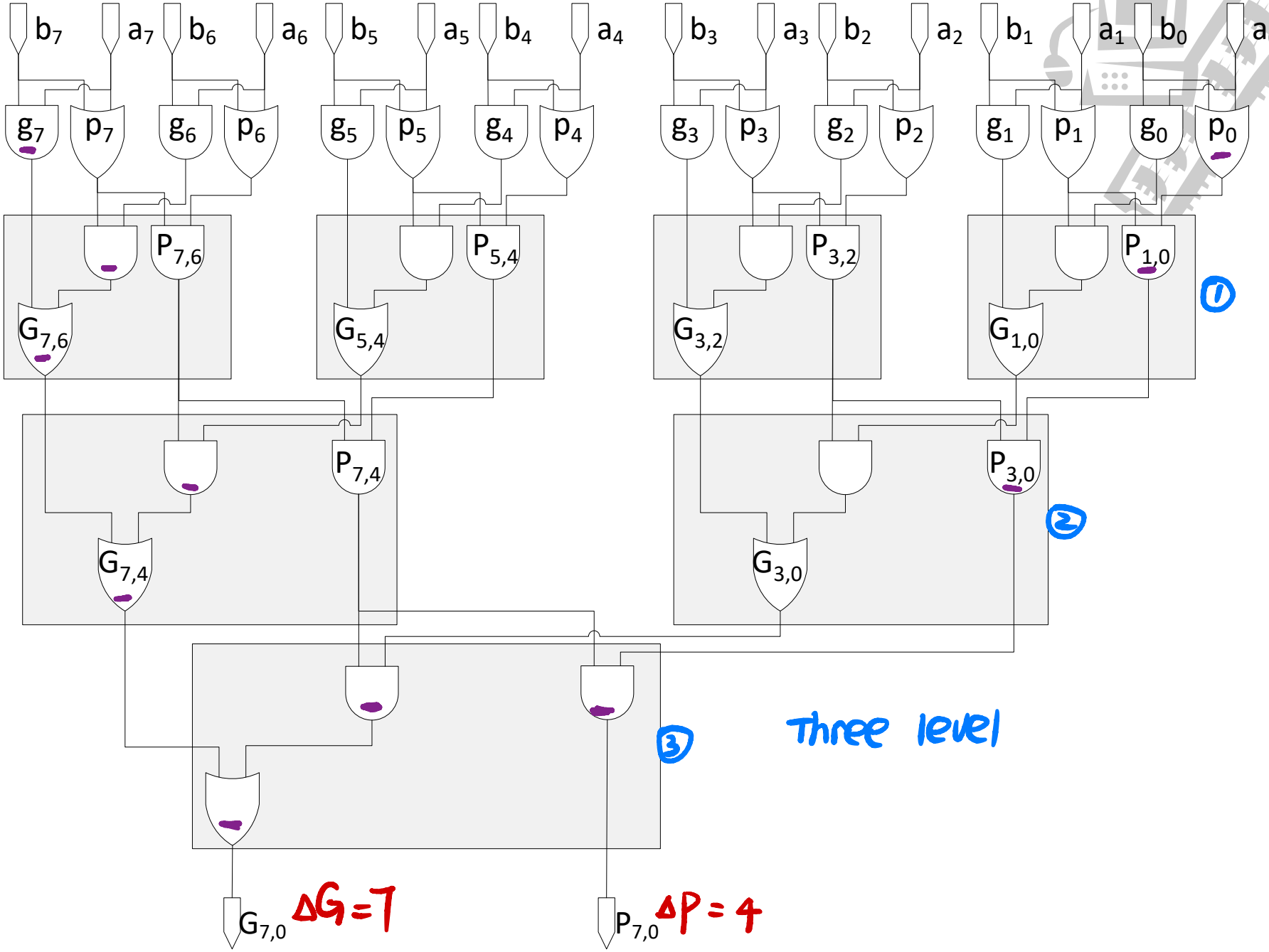

b7
a7
b6
a6
b5
a5
b4
a4
b3
a3
b2
a2
b1
a1
b0
a0
g7
p7
g6
p6
g5
p5
g4
p4
g3
p3
g2
p2
g1
p1
g0
p0



the output of this shaded block : Two bits group signals Propergate ($P_{1,0}$) and generate ($G_{1,0}$)



same circuit is used to generate the 4 bit signals.


b7 a7 b6 a6 b5 a5 b4 a4 b3 a3 b2 a2 b1 a1 b0 a0
g7 p7 g6 p6 g5 p5 g4 p4 g3 p3 g2 p2 g1 p1 g0 p0
P7,6 P5,4 P3,2 P1,0
G7,6 G5,4 G3,2 G1,0
1
P7,4 P3,0
G7,4 G3,0
2
3
Three level
G7,0 ΔG=7
P7,0 ΔP=4

Do Delay analysis

Generate/Propagate Equations

first level of logic has one gate: "AND" "OR" gate

Depth(P,G) (1, 1) (2, 3) (3, 5) (4, 7)

$g_0 = a_0 b_0$

$p_0 = a_0 + b_0$

$g_1 = a_1 b_1$

$p_1 = a_1 + b_1$

$g_2 = a_2 b_2$

$p_2 = a_2 + b_2$

$g_3 = a_3 b_3$

$p_3 = a_3 + b_3$

$g_4 = a_4 b_4$

$p_4 = a_4 + b_4$

$g_5 = a_5 b_5$

$p_5 = a_5 + b_5$

$g_6 = a_6 b_6$

$p_6 = a_6 + b_6$

$g_7 = a_7 b_7$

$p_7 = a_7 + b_7$

so the depth of "P" signal come out of the 1st level is one "OR" and one "AND"

"AND" & "OR" for the generate

$G_{1,0} = g_1 + p_1 g_0$

$P_{1,0} = p_1 p_0$

one "AND" for the propagate

So if just counting gate, then the generate signal will have a depth of 2. Propagate signal has depth of 2.

$G_{3,2} = g_3 + p_3 g_2$

$P_{3,2} = p_3 p_2$

$G_{5,4} = g_5 + p_5 g_4$

$P_{5,4} = p_5 p_4$

$G_{7,6} = g_7 + p_7 g_6$

$P_{7,6} = p_7 p_6$

depth of G: +2

Two more level

$G_{3,0} = G_{3,2} + P_{3,2} G_{1,0}$

$P_{3,0} = P_{3,2} P_{1,0}$

one more "AND" gate for propagate signal

depth of P: +1

$G_{7,4} = G_{7,6} + P_{7,6} G_{5,4}$

$P_{7,4} = P_{7,6} P_{5,4}$

$G_{7,0} = G_{7,4} + P_{7,4} G_{3,0}$

$P_{7,0} = P_{7,4} P_{3,0}$

Delays in the GP "Parallel Prefix"

表示 depth

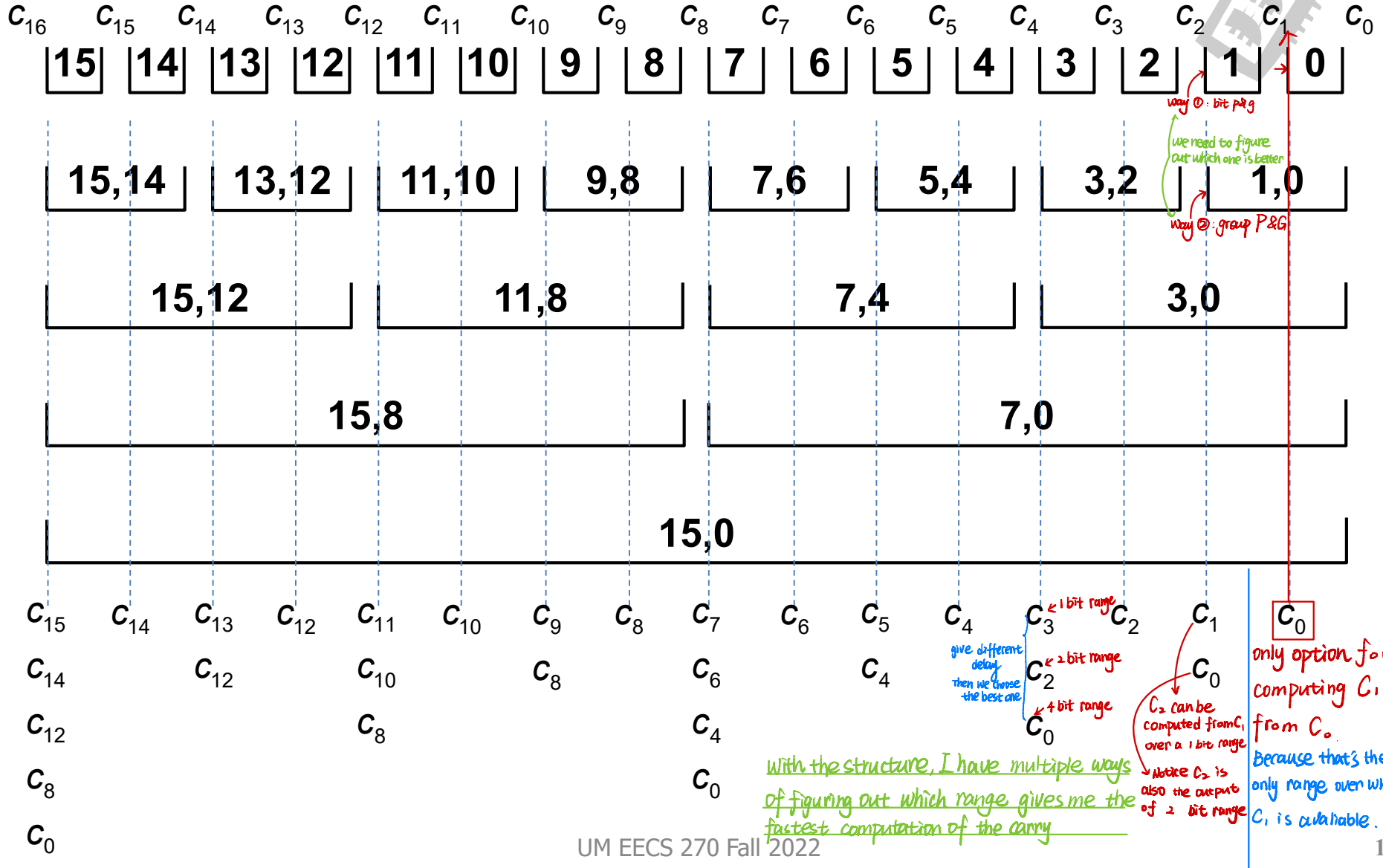
How many gates do we have at the end of that range

Level	Bit-Range Width	Δ_P	Δ_G
1	1	1	1
2	2	2	3
3	4	3	5
4	8	4	7
5	16	5	9

Analyze when the carriers become available

I need every carries for the sum.
eg: I don't need C_1 for C_2 but I need C_1 for sum.

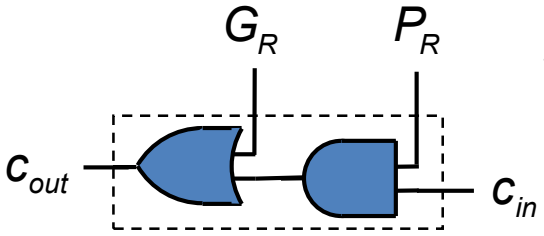
Options for Carry Computation



With the structure, I have multiple ways of figuring out which range gives me the fastest computation of the carry

Carry equations

Level	Bit-Range Width	Δ_P	Δ_G
1	1	1	1
2	2	2	3
3	4	3	5
4	8	4	7
5	16	5	9

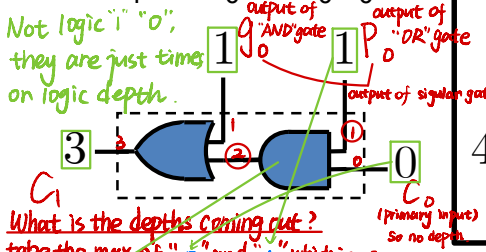


	Δ	c_{in}
c_0	0	
c_1	3	c_0
c_2	4	c_0
c_3	6	c_2
c_4	6	c_0
c_5	8	c_4
c_6		
c_7		
c_8		

computed from

c_3 and c_4 are appearing at the same time because c_4 is

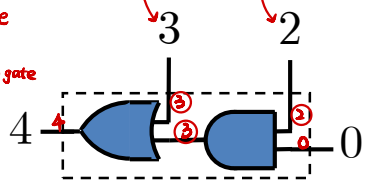
$c_1 = g_0 + p_0 c_0$



What is the depths coming out?
take the max of "0" and "1" which is 1.
then plus 1 which is "2", then the max between "2" and "1" is "2" 2+1=3.

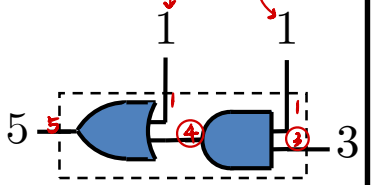
better over two bit range

$c_2 = G_{1,0} + P_{1,0} c_0$

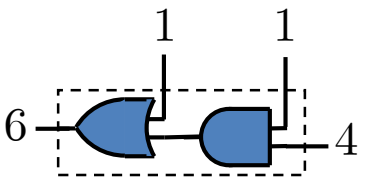


over the one bit range

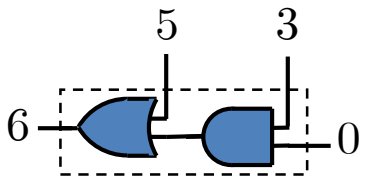
$c_2 = g_1 + p_1 c_1$



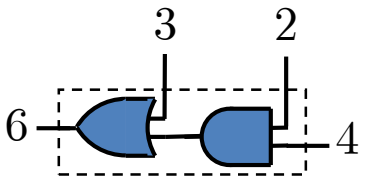
$c_3 = g_2 + p_2 c_2$



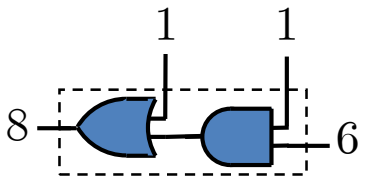
$c_4 = G_{3,0} + P_{3,0} c_0$



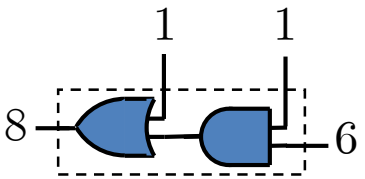
$c_4 = G_{3,2} + P_{3,2} c_2$



$c_4 = g_3 + p_3 c_3$

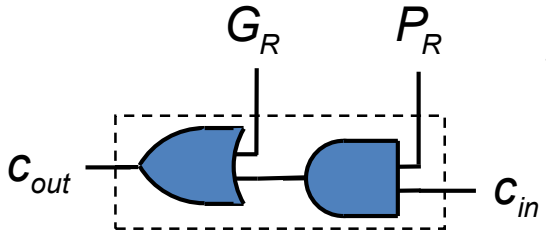


$c_5 = g_4 + p_4 c_4$



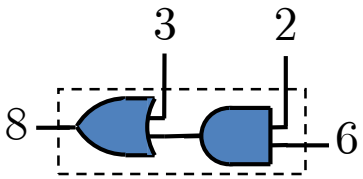
Carry equations

Level	Bit Range Width	Δ_P	Δ_G
1	1	1	1
2	2	2	3
3	4	3	5
4	8	4	7
5	16	5	9

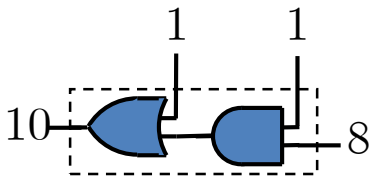


	Δ	c_{in}
c_0	0	
c_1	3	c_0
c_2	4	c_0
c_3	6	c_2
c_4	6	c_0
c_5	8	c_4
c_6	8	c_4
c_7	10	c_6
c_8	8	c_0

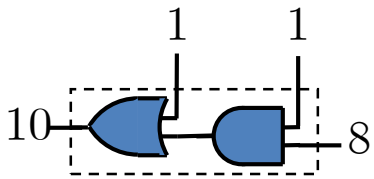
$c_6 = G_{5,4} + P_{5,4}c_4$



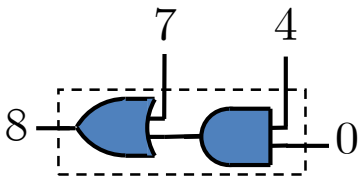
$c_6 = g_5 + p_5c_5$



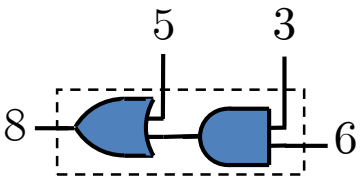
$c_7 = g_6 + p_6c_6$



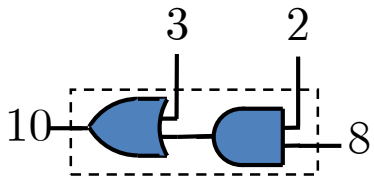
$c_8 = G_{7,0} + P_{7,0}c_0$



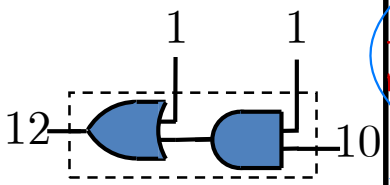
$c_8 = G_{7,4} + P_{7,4}c_4$



$c_8 = G_{7,6} + P_{7,6}c_6$

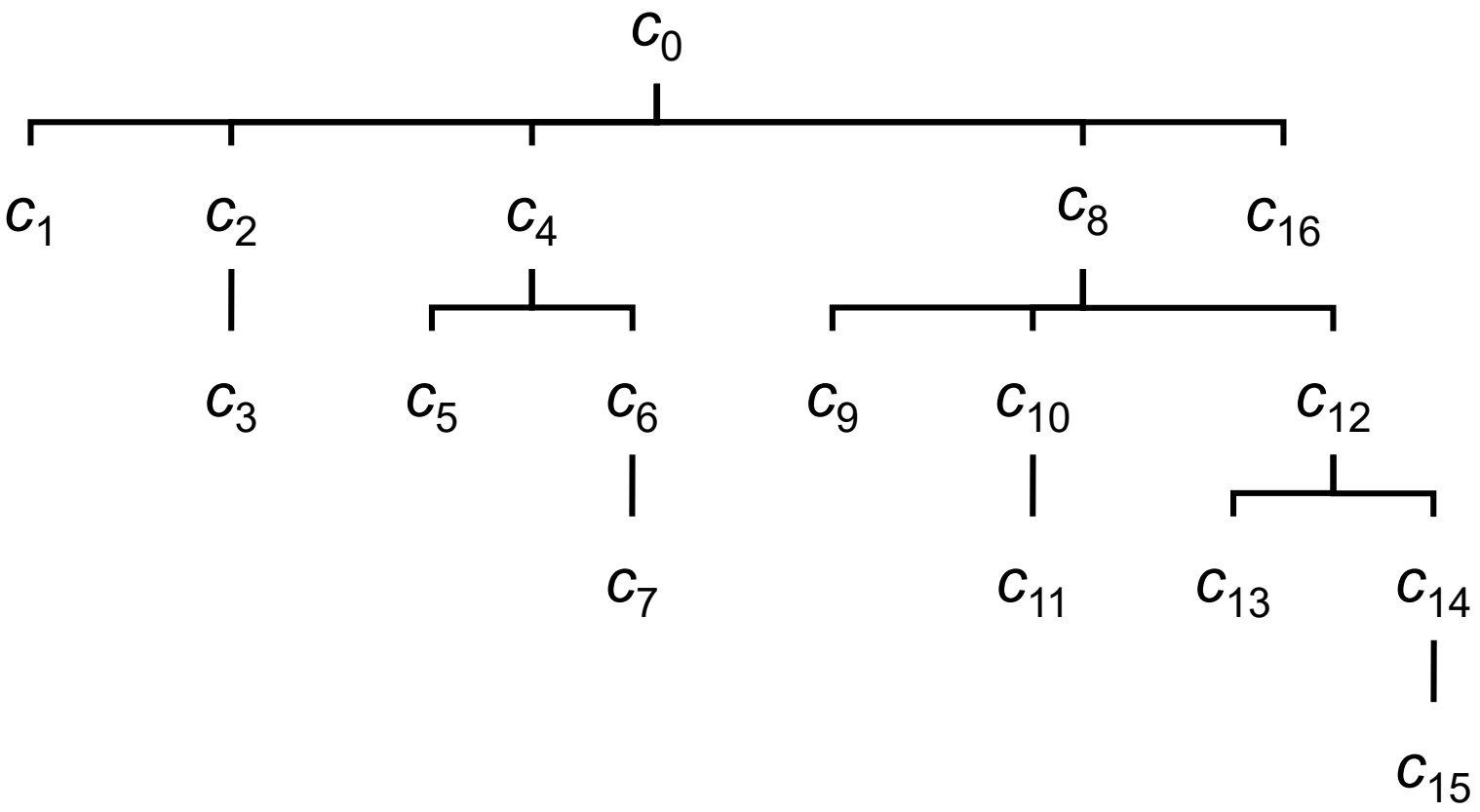


$c_8 = g_7 + p_7c_7$

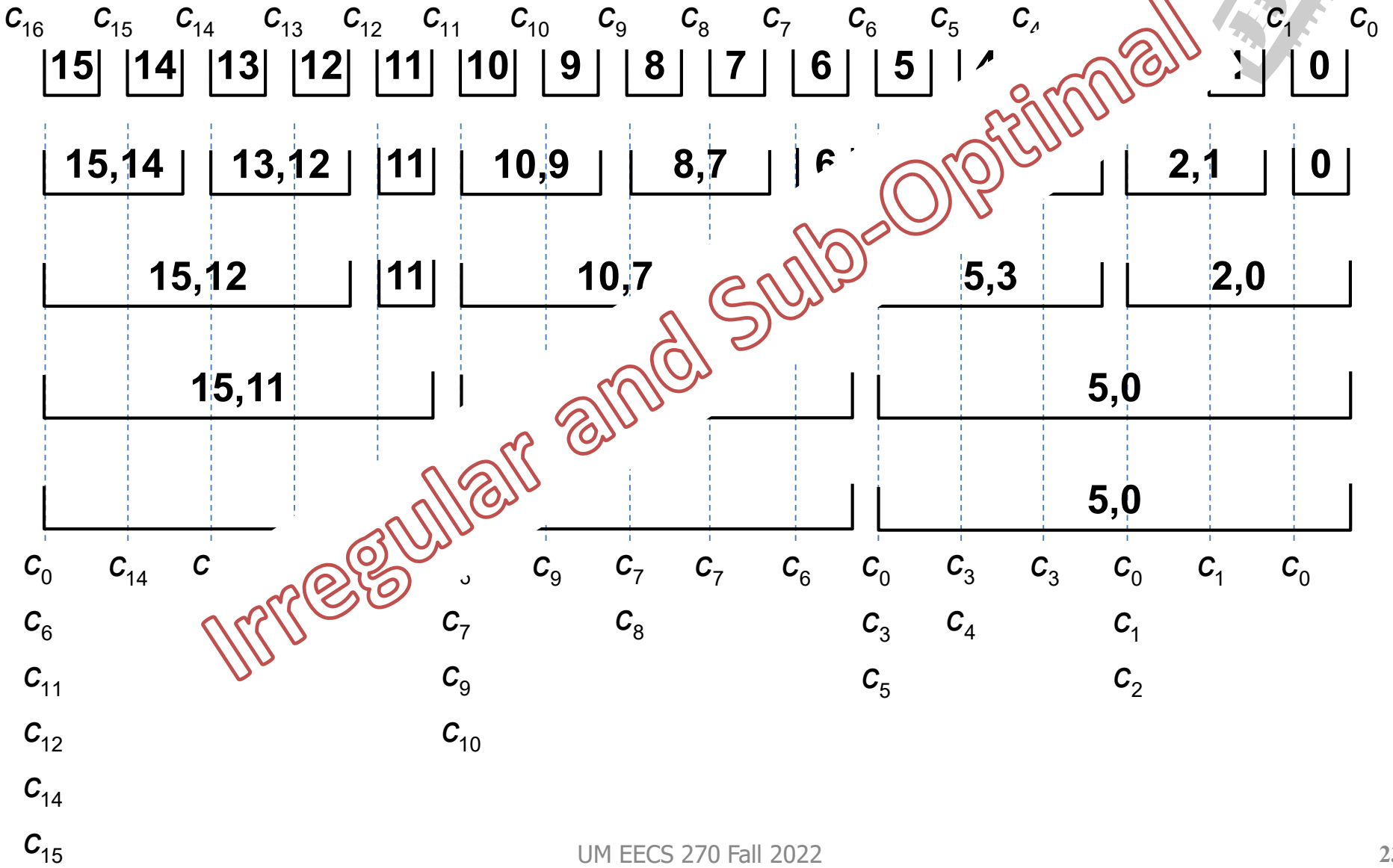


the depth over here is not increasing linearly
↳ If this is the overflow bit, it actually appear earlier than the carry in to the last bit.

Carry Dependency Tree
(which carry propagates to which other carry)



Here's a different way to create groups

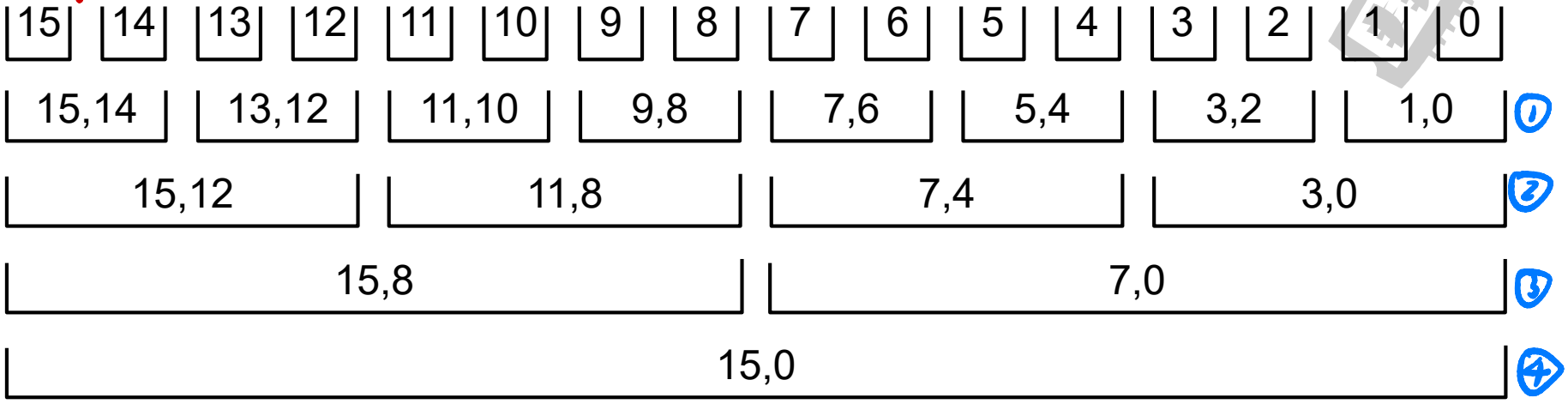


Irregular and Sub-Optimal

We can start by ranges that are over four bits instead of two bits

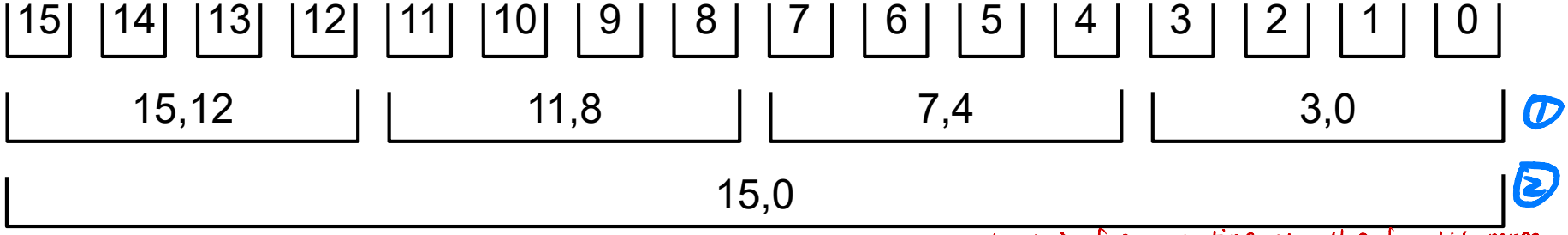
Depth of GP Structure

① grouping into 2



$$d = \log_2 n = \log_2 16 = 4$$

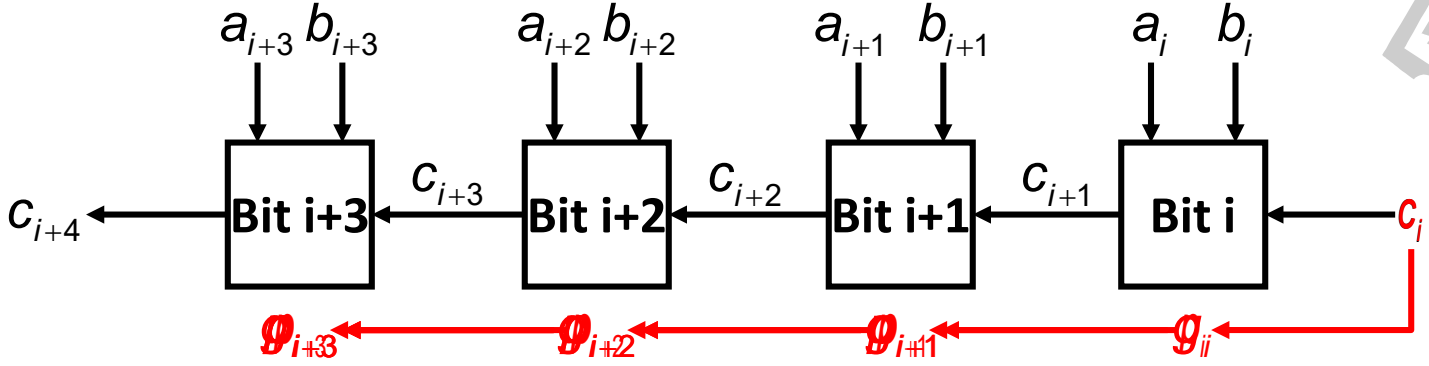
② grouping into 4.



$$d = \log_4 n = \log_4 16 = 2$$

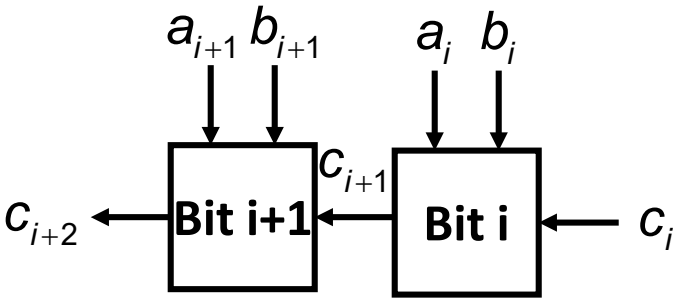
- the logic for computing over the four bit range is more complex
- the ripple structure is when the range is 1 bit.

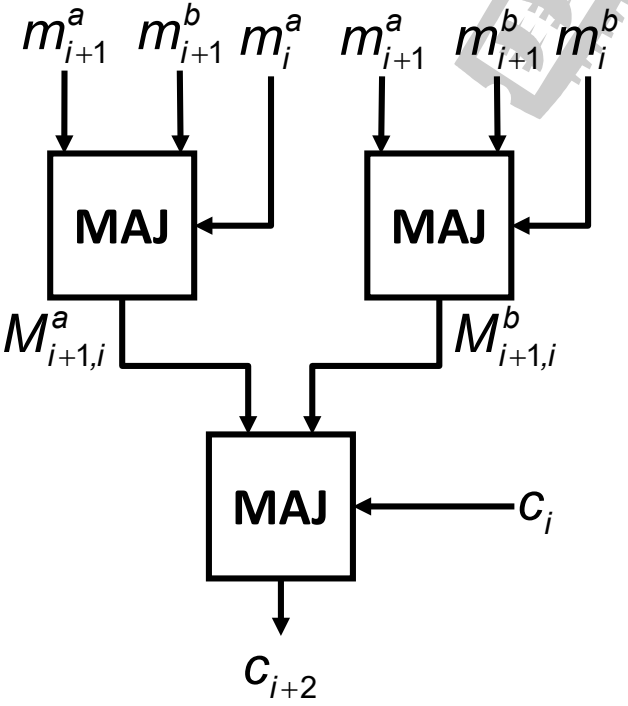
Group Generate & Propagate: 4 Bits



$$
\begin{aligned}
c_{i+4} &= g_{i+3} + p_{i+3}c_{i+3} \\
&= g_{i+3} + p_{i+3}\left(g_{i+2} + p_{i+2}c_{i+2}\right) \\
&= g_{i+3} + p_{i+3}g_{i+2} + p_{i+3}p_{i+2}c_{i+2} \\
&= g_{i+3} + p_{i+3}g_{i+2} + p_{i+3}p_{i+2}\left(g_{i+1} + p_{i+1}c_{i+1}\right) \\
&= g_{i+3} + p_{i+3}g_{i+2} + p_{i+3}p_{i+2}g_{i+1} + p_{i+3}p_{i+2}p_{i+1}c_{i+1} \\
&= g_{i+3} + p_{i+3}g_{i+2} + p_{i+3}p_{i+2}g_{i+1} + p_{i+3}p_{i+2}p_{i+1}\left(g_i + p_i c_i\right) \\
&= \underbrace{g_{i+3} + p_{i+3}g_{i+2} + p_{i+3}p_{i+2}g_{i+1} + p_{i+3}p_{i+2}p_{i+1}g_i}_{G_{i+3,i}} + \underbrace{p_{i+3}p_{i+2}p_{i+1}p_i c_i}_{P_{i+3,i}}
\end{aligned}
$$

Another Variation: M&M Structure





$$\mathrm{MAJ}(x,y,z) \equiv xy + xz + yz$$

$$c_{i+2} = \mathrm{MAJ}(a_{i+1}, b_{i+1}, c_{i+1})$$

$$= \mathrm{MAJ}(a_{i+1}, b_{i+1}, \mathrm{MAJ}(a_i, b_i, c_i))$$

⇓ Algebra

$$= \mathrm{MAJ}(\mathrm{MAJ}(a_{i+1}, b_{i+1}, a_i), \mathrm{MAJ}(a_{i+1}, b_{i+1}, b_i), c_i)$$

Let $m^a_i \equiv a_i, m^b_i \equiv b_i,$ etc.

$$\therefore c_{i+2} = \mathrm{MAJ}(\underbrace{\mathrm{MAJ}(m^a_{i+1}, m^b_{i+1}, m^a_i)}_{M^a_{i+1,i}}, \underbrace{\mathrm{MAJ}(m^a_{i+1}, m^b_{i+1}, m^b_i)}_{M^b_{i+1,i}}, c_i)$$

How are we compute the sum

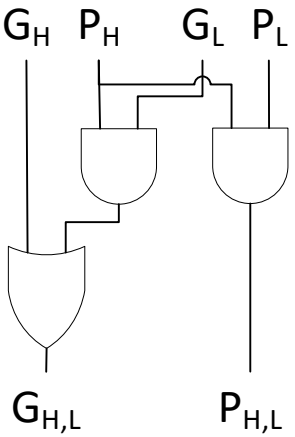
CLA Adder Building Blocks

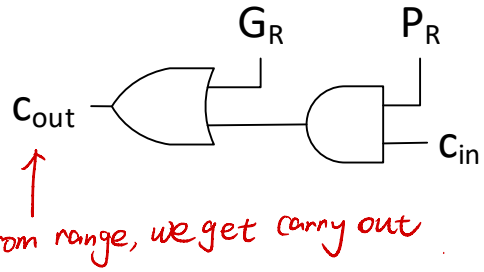
high part

law part

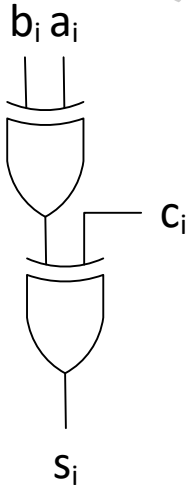
combine them into a range

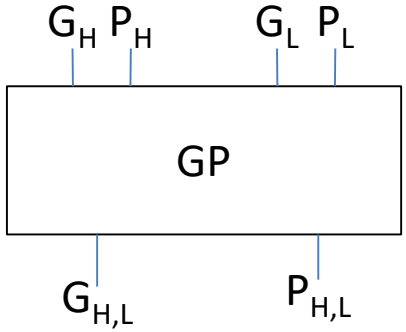
compute the group signal

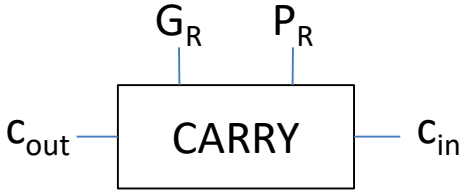


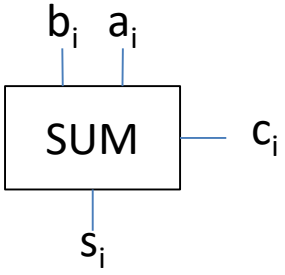


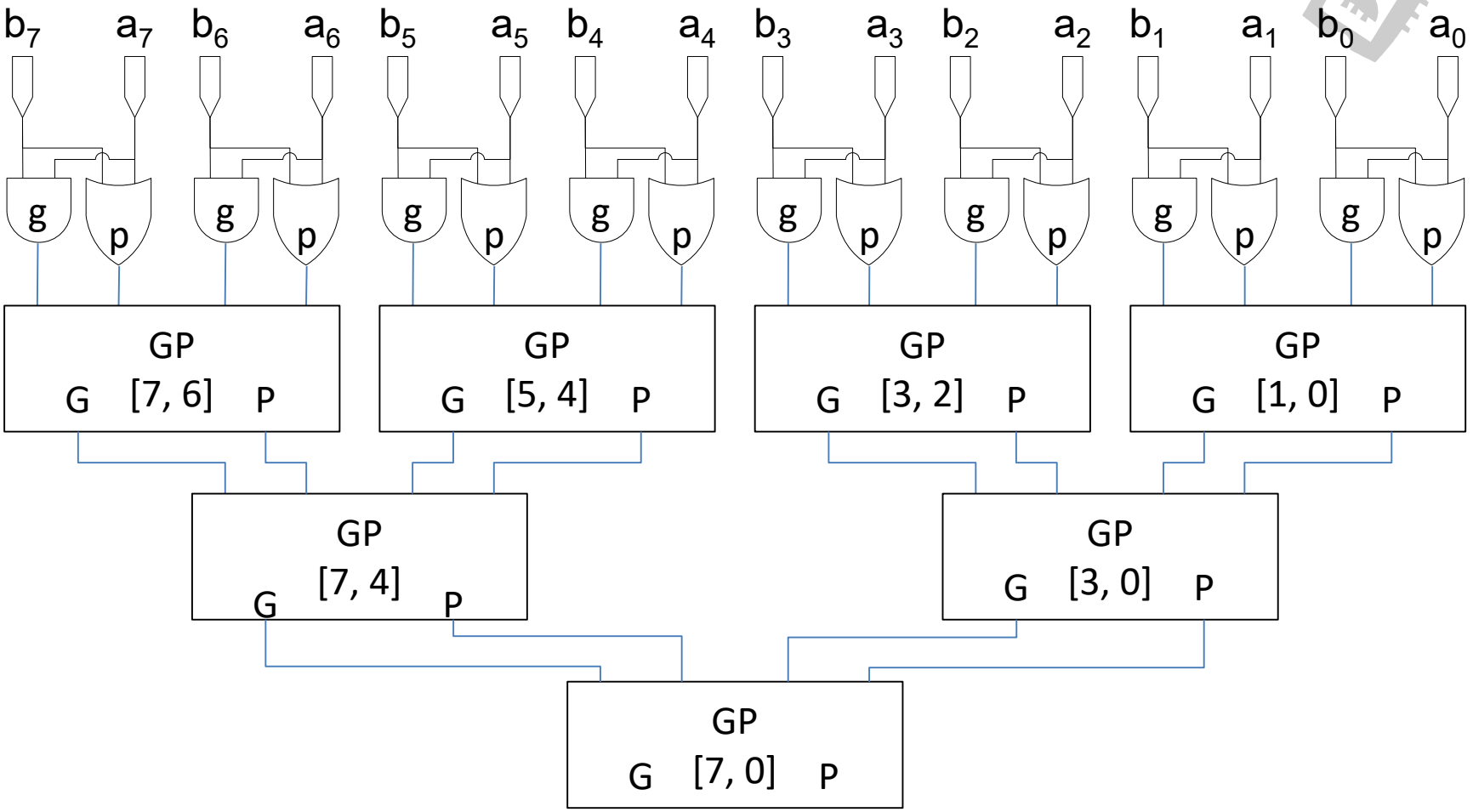
from range, we get carry out

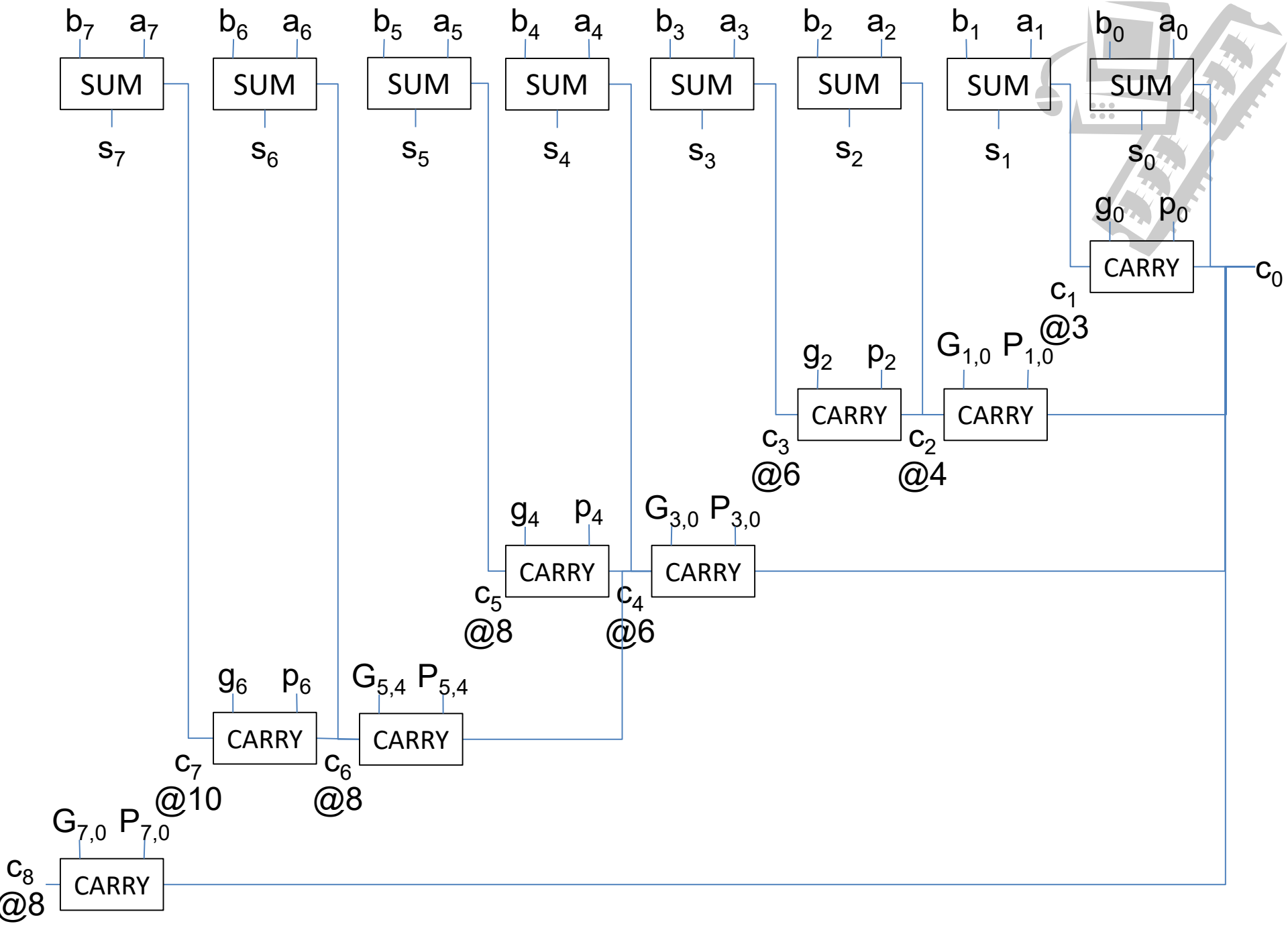




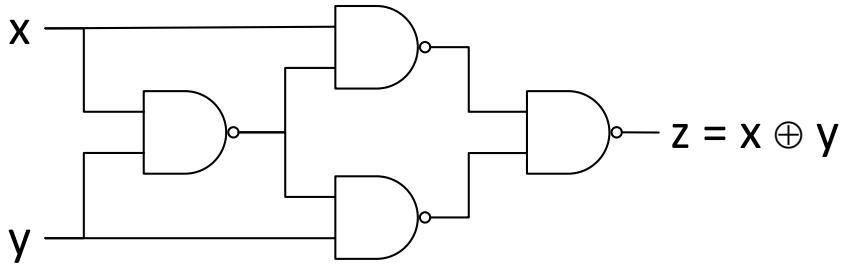





b7 a7 b6 a6 b5 a5 b4 a4 b3 a3 b2 a2 b1 a1 b0 a0
g p g p g p g p g p g p g p g p
GP G [7, 6] P
GP G [5, 4] P
GP G [3, 2] P
GP G [1, 0] P
GP [7, 4] G P
GP G [3, 0] P
GP G [7, 0] P


b7 a7
b6 a6
b5 a5
b4 a4
b3 a3
b2 a2
b1 a1
b0 a0
SUM
s7
s6
s5
s4
s3
s2
s1
s0
g0 p0
CARRY
c0
c1
@3
g2 p2
G1,0 P1,0
c3
@6
c2
@4
g4 p4
G3,0 P3,0
c5
@8
c4
@6
g6 p6
G5,4 P5,4
c7
@10
c6
@8
G7,0 P7,0
c8
@8

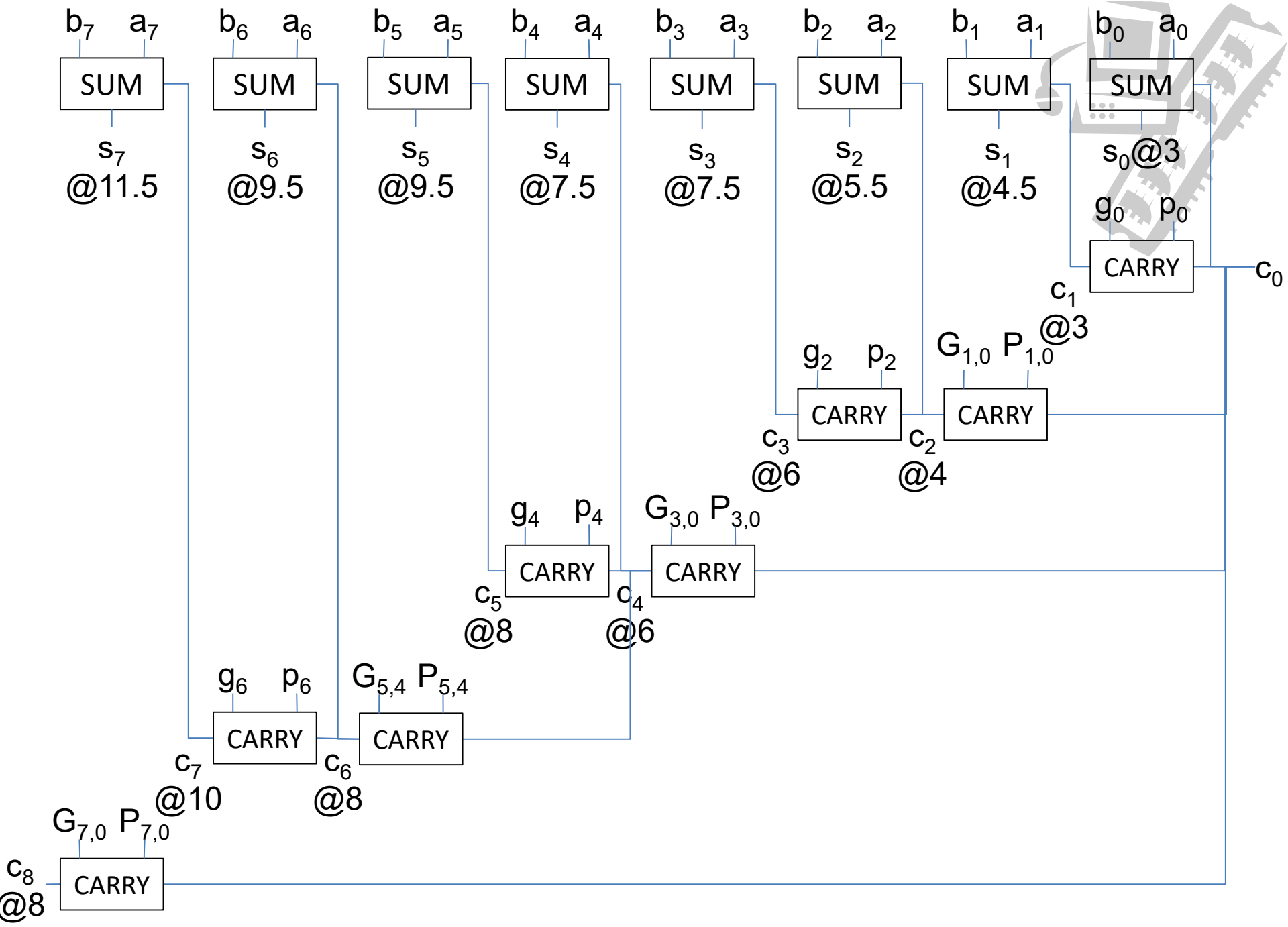
Slightly More Accurate Timing Analysis



$$t_p^{AND} = t_p^{NAND} + t_p^{INV}$$

$$\text{Assume } t_p^{NAND} = t_p^{INV} = 0.5$$

$$\therefore\ t_p^{XOR} = 1.5$$


b7 a7
SUM
s7
@11.5
b6 a6
SUM
s6
@9.5
b5 a5
SUM
s5
@9.5
b4 a4
SUM
s4
@7.5
b3 a3
SUM
s3
@7.5
b2 a2
SUM
s2
@5.5
b1 a1
SUM
s1
@4.5
b0 a0
SUM
s0@3
g0 p0
CARRY
c0
c1
@3
g2 p2
CARRY
G1,0 P1,0
CARRY
c3
@6
c2
@4
g4 p4
CARRY
G3,0 P3,0
CARRY
c5
@8
c4
@6
g6 p6
CARRY
G5,4 P5,4
CARRY
c7
@10
c6
@8
G7,0 P7,0
CARRY
c8
@8

Summary: Fast Addition

Rich Literature

- Carry Skip
- Carry Save
- Carry Select
- Kogge–Stone Adder
- Brent–Kung Adder
- etc.